Fall 10-1-1979

The Multigrid Iteration Applied to the Collocation Method ; CU-CS-164-79

John Gary
University of Colorado Boulder

THE MULTIGRID ITERATION APPLIED TO THE COLLOCATION METHOD

by

John Gary
Computer Science Department
University of Colorado
Boulder, Colorado 80309

CU-CS-164-79 October, 1979

This material is based upon work supported by the National Science Foundation under Grant No. ENG77-26893.

THE MULTIGRID ITERATION APPLIED TO THE COLLOCATION METHOD

by

John Gary
Computer Science Department
University of Colorado
Boulder, Colorado 80309

CU-CS-164-79 October, 1979

This material is based upon work supported by the National Science Foundation under Grant No. ENG77-26893.

THE MULTIGRID ITERATION APPLIED TO THE COLLOCATION METHOD

by

John Gary
Computer Science Department
University of Colorado
Boulder, Colorado 80309

This material is based upon work supported by the National Science Foundation under Grant No. ENG77-26893.

Abstract

We describe an application of the multigrid iteration to the collocation approximation for elliptic equations. A block relaxation and a projection operator specific to the collocation approximation were required to obtain convergence. We illustrate the method with the two point boundary problem. Results are also given for an elliptic problem in two dimensions. Comments on computational efficiency are given along with operational counts.

1. Introduction

We are concerned here with the solution of collocation approximations to linear boundary value problems, either two point boundary value problems or elliptic problems in two dimensions. Our objective is to demonstrate that the multigrid iteration can be applied to these approximations in much the same way as it is applied to finite difference approximations with two minor changes. The multiqrid iteration is taken from the work of Brandt [2]. A block Gauss Seidel smoothing is used since point relaxation does not converge for the collocation approximation. Parter has also used a block relaxation for the collocation approximation and provides a proof of convergence for the one-dimensional case [1]. The second difference lies in projection of the residue from the fine to the course mesh. The convergence rate of the multigrid for collocation is faster than for finite differences. However, the computational work required to set up the collocation equations is very great. Unless the collocation coefficients can be stored, the method is probably not competitive with finite difference methods. Our work will be based on the cubic Hermite basis in one dimension and the bicubic Hermite basis in two dimensions [4].

2. Application to a two point boundary value problem

In this section we will illustrate the multigrid iteration as it applies to the following boundary problem.

$$a(x)\, u_{xx} + b(x)\, u_x + c(x)\, u = f(x) \qquad (2\text{-}1)$$

The boundary conditions are the following.

$$u(0) = A \qquad u(1) = B$$

The multigrid method is of little value for such one dimensional problems since a direct solution is quite efficient. We use the one

dimensional problem as a simple "test bed" for the method, and also to simplify this presentation. The collocation method is based on an approximation of the solution $u(x)$ by a linear combination of basis functions. The differential equation and the boundary conditions are then used to determine the coefficients in this representation.

We use the Hermite cubic functions as the basis. These are piecewise cubic polynomials on each mesh subinterval $[x_{i-1}, x_i]$. There are two such polynomials associated with each mesh point which are non zero only in the single interval to each side of the mesh point. These polynomials are pieced together so that the composite basis function has a continuous first derivative. We denote the Hermite basis functions on the interval $[-1, 1]$ by $\psi(x)$ and $\chi(x)$. These functions have the following properties.

$$\psi(0) = 1 \quad \psi'(0) = 0 \quad \chi(0) = 0 \quad \chi'(0) = 1$$

The basis functions associated with each grid point x_i are denoted by $\phi_i^s(x)$

$$\phi_i^1(x) = \psi((x-x_i)/h) \quad \phi_i^2(x) = \chi((x-x_i)/h)\ h$$

We use an equally spaced mesh $x_i = i/N$ where $1 \le i \le N$. The collocation approximation of the solution $u(x)$ is denoted by $U(x)$ where

$$U(x) = \sum_{i=0}^{N} c_i^1\, \phi_i^1(x) + c_i^2 \phi_i^2(x) \tag{2-2}$$

The unknown coefficients $\{c_i^s\}$ are determined by the requirement that the differential equation or boundary conditions be satisfied at the collocation points.

These collocation points will be denoted by z_i^s for $0 \le i \le N$ and $1 \le s \le 2$ where

$$z_i^1 = x_i - d\ h \quad z_i^2 = x_i + d\ h \quad \text{for } 1 \le i \le N-1 \tag{2-3}$$

The parameter d is determined so that these collocation points are the Gaussian quadrature points on the mesh interval $[x_{i-1},x_i]$ and thus has the value $d = (\sqrt{3} - 1)/(2\sqrt{3})$. At the boundary the collocation points are $z_0^1 = 0 \quad z_0^2 = d\,h \quad z_N^1 = 1-d\,h \quad z_N^2 = 1$

We require that U(x) satisfy the differential equation at each interior collocation point. At the interior point z_i^1 which lies to the left of the mesh point x_i this substitution yields the following equation

$$\sum_{j=i-1}^{i} \sum_{s=1}^{2} [a(z_i^1)\,\phi_{jxx}^s(z_i^1) + b(z_i^1)\,\phi_{jx}^s(z_j^1) + c(z_i^1)\,\phi_j^s(z_i^1)]\,c_j^s = f(z_i^1) \qquad (2\text{-}4)$$

We also require that U(x) satisfy the boundary conditions at the points z_0^1 and z_N^2. Therefore we have 2N+2 equations for the same number of unknowns $\{c_i^s\}$. To determine the coefficients $\{c_i^s\}$ we must solve the following linear system of equations

$$M\underline{c} = \underline{f} \qquad (2\text{-}5)$$

where $\underline{c}$ and $\underline{f}$ are vectors

$$\underline{c} = (c_0^1, c_0^2, \ldots c_N^2)$$
$$\underline{f} = (f(z_0^1), f(z_0^2), \ldots f(z_N^2))$$

and the matrix M can be written in block form as

$$M = \begin{vmatrix} B_0 & C_0 & . & . & . \\ A_1 & B_1 & C_1 & . & . \\ . & . & . & . & . \\ . & . & . & A_N & B_N \end{vmatrix} \qquad (2\text{-}6)$$

where the blocks A_i, B_i, and C_i are 2×2 matrices. Note that the matrix M is neither symmetric or diagonally dominant. Therefore standard iterative methods may not be successful with this problem. Boley and Parter have obtained convergence results for block SOR applied to (2-5) [1].

The blocks are the 2×2 matrices used to define M in (2-5). The multigrid method which we use is also based on a block Gauss-Seidel iteration. This insures that the relaxation method used as a part of the multigrid iteration is a smoothing operator in the sense used by Achi Brandt [2]. A distributed relaxation method might be devised for this problem, however, it would probably be no more efficient than the block method provided the inverse of the 2×2 matrices needed in the Gauss-Seidel iteration can be computed once and then saved.

3. The multigrid iteration

We will assume that the reader is familiar with the multigrid method. The basic idea is to use a sequence of mesh refinements, in most cases each refinement has half the mesh spacing of the preceding refinement. A smoothing iteration such as Gauss-Seidel is used to reduce error in the higher frequencies. After a few iterations, the higher frequency error will be largely eliminated, and then the error reduction rate of the Gauss-Seidel will drop, since the iteration is not very efficient in eliminating low frequency error. At this point a projection to the next mesh is made and the process continues on the course mesh. The low frequency errors will project into higher frequency errors on the courser meshes so that there is hope for a more rapid convergence. We write the equation to be solved on the mesh with spacing h as follows (i.e., we rewrite (2-5)).

$$M_h \underline{U}_h = \underline{F}_h$$

We also change to use of U_h to denote the collocation coefficients on the h mesh. To denote the projection from the fine h mesh

to the course 2h mesh we use I_h^{2h}. The injection from the course mesh is denoted by I_{2h}^h.

We use a block Gauss-Seidel iteration to smooth the errors on each mesh. In terms of the block matrices this iteration is defined by

$$B_i U_i^{\nu+1} = F_i - A_i U_i^{\nu+1} - C_i U_i^{\nu} \tag{3-2}$$

Here $U^{\nu+1}$ denotes the successive iterates and U_i^{ν} is used for the two coefficients at the point x_i. We use the "full approximation storage" algorithm (FAS) as described by Brandt [2]. The "cycle C" algorithm would also work and is likely to be more efficient for linear problems, however, the FAS algorithm is more general, so we tested it. In the FAS case the residue $\underline{F}_h - M_h \underline{U}_h$ is projected from the fine to the course mesh according to the following equation

$$\underline{F}_{2h} = M_{2h}(I_h^{2h}(\underline{U}_h)) + \hat{I}_h^{2h}(\underline{F} - M_h(\underline{U}_h)) \tag{3.3}$$

The iteration on the course 2h mesh is applied to

$$M_{2h}\underline{U}_{2h} = \underline{F}_{2h} \tag{3.4}$$

The definition of the projection operator $\hat{I}_h^{2h}$ for the residue requires a longer explanation which we give below. The projection of the solution simply involves taking the course mesh value to be equal to the fine mesh value at the same mesh point, that is take the values at every other fine mesh point down to the course mesh. To move the solution from the course to the fine mesh Brandt suggests the formula

$$U_h = U_h^o + I_{2h}^h (U_{2h} - I_h^{2h}(U_h^o)) \tag{3.5}$$

The operator I_{2h}^h is defined by linear interpolation, that is the values on the fine mesh midpoints are obtained by average of the values at the two course mesh points on either side.

The proper definition of the projection of the residue from the fine to course mesh gave us some difficulty. The use of simple linear interpolation from the fine to course mesh seemed to yield a divergent iteration. Following a suggestion of Brandt, we regard $\hat{I}_h^{2h}$ as distributing residue from each fine mesh point to the surrounding course mesh points [5]. This yields a type of linear interpolation with positive weights, but provides a rational for choosing the weights. In Figure 1 we show the fine and course mesh with the fine mesh shown above the course in order to clarify the figure. The residue on the fine mesh points a, b, c, d is distributed to the course mesh points A, B, C as indicated by the arrows. Each fine mesh point gives residue to the two closest course mesh points. The residue contributed to the course mesh point is proportional to the distance to the point divided by the sum of the distances to the two course mesh points. We denote the residue contributed by the point a to A and B by $r_{a,A}$ and $r_{a,B}$. The residue at a is given by r_a and that at A by r_A. The distance between points a and A is denoted by $D_{a,A}$ and between a and B by $D_{a,B}$. The residue contributed by a to A and B is then given by the relations

$$
\begin{aligned}
r_{a,A} &= \frac{.5\, D_{a,B}\, r_a}{D_{a,A} + D_{a,B}} = .5(1-\alpha)\, r_a \qquad \alpha = D_{a,A} / (D_{a,A}+D_{a,B}) \\
r_{a,B} &= \frac{.5\, D_{a,A}\, r_b}{D_{a,A} + D_{a,B}} = .5\, \alpha\, r_a \qquad (3\text{-}6) \\
D_{a,A} &= 1-3\beta \qquad \beta = (\sqrt{3}-1 / 2\sqrt{3} \\
D_{a,B} &= 1-\beta
\end{aligned}
$$

Since there are twice as many fine mesh points as course points, each fine mesh point gives half of its residue to the course mesh. Note that

if the mesh spacing h is unity, then the distance between the collocation points and the nearest mesh point is $\beta = (\sqrt{3}-1) / 2\sqrt{3}$. The total residue projected from the fine mesh to the course mesh point B is given by

$$r_B = .5\,\alpha\, r_a + .5(1-\alpha)\, r_b + .375\, r_c + .125\, r_d \qquad (3\text{-}7)$$

At the boundary the projection is as indicated in Figure 2. The residue projected to the course points A and B are

$$r_A = r_a$$

$$r_B = .5\, r_b + .5(1-\alpha)\, r_c + .5\,\alpha\, r_d \qquad (3\text{-}8)$$

The weights for each course mesh point sum to one, therefore a constant residue projects into the same constant. If the weights did not sum to the same value at all the course mesh points, then there would be a distortion of the low frequencies which would probably have a bad effect on the multigrid iteration. The boundary points are an exception. Since the boundary condition is satisified exactly at all the mesh levels, the residue is zero at these boundary points. Note that the sum of the residue over the mesh is preserved by the projection to the course mesh.

4. The results for the two point boundary value problem.

We next display the results for the multigrid iteration applied to the following problem.

$$a(x)\, u' + a'(x)\, u' = f(x) \qquad (4\text{-}1)$$

In this case we use $a(x) = e^{-x}$ and we choose $f(x)$ so that $u(x) = \sin \pi x$ is the solution of the differential equation. We use a Dirichlet boundary condition at the left and a Neumann condition at the right, that is

$$u(0) = 0 \qquad u'(1) = -\pi \qquad (4\text{-}2)$$

The mesh used for this experiment contained 33 points, that is N = 33.

There are 5 successive grid refinements containing 3, 5, 9, 17 and 33 points. We used a fixed iteration scheme, each cycle of which involved one Gauss-Seidel iteration on mesh levels 5, 4, and 3, two iterations on level 2 and three iterations on level 1 (level 5 is the finest level). One cycle starts with the iteration on level 4, drops down to level 1 and proceeds back thru level 5. One work unit is defined to be the work required for one Gauss-Seidel iteration on the finest mesh, therefore an iteration on level 4 requires .5 work units, and an iteration on level one requires 1/16 units. One cycle requires 3.19 units. For a two dimensional problem the work units are more heavily dominated by the work required on the finest mesh.

The results are displayed in Table I. The norm used to measure the difference between iterates $\|U^{\nu+1} - U^{\nu}\|$ involves both the function values and the derivaties at each mesh point. The L_2 norm is used, that is

$$\|U^{\nu}\| = \left(\sum_i U(x_i)^2 + U'(x_i)^2\right)^{1/2} \tag{4-3}$$

The errors shown are the L_2 norm for the function values

$$e(u) = \left(\sum_i (U(x_i) - u(x_i))^2\right)^{1/2} \tag{4-4}$$

and the L_2 norm for the derivatives.

$$e(u') = \left(\sum_i (U'(x_i) - u'(x_i))^2\right)^{1/2} \tag{4-5}$$

Note that the coefficients c_i^s in the collocation representation of the solution are just the values of the solution and its derivative at each mesh point, that is

$$c_i^1 = U(x_i) \qquad c_i^2 = U'(x_i)$$

The average convergence rate per cycle, for the difference in iterates averaged over 5 cycles is 0.28. It is possible to obtain an estimate for this convergence rate for a constant coefficient problem ($a(x) = 1$) having periodic boundary conditions. The problem can then be solved by Fourier analysis as described by Brandt [2]. In our case this requires computation of the eigenvalues of a 2×2 matrix because of the block nature of the iteration. The Fourier analysis involves a finite number of frequencies equal to the number of mesh points. The smoothing rate defined by Brandt is the convergence rate for the upper half of these Fourier modes, that is the higher frequency modes. This smoothing rate has the value $\underline{\mu} = 0.43$. The convergence rate per work unit is then given by $\underline{\mu}^{1-2^{-d}}$ where d is the dimension (d=1). This value is 0.26 which is in good agreement with the observed value of 0.28.

5. Application to an elliptic equation in two dimensions

In this section we will extend the collocation-multigrid approximation to the following elliptic equation in two dimensions.

$$a_1(x,y)u_{xx} + a_2(x,y)u_{yy} + a_3(x,y)u_x + a_4(x,y)u_y + a_5(x,y) = f(x,y) \qquad (5\text{-}1)$$

for $0 \le x,\ y \le 1$

The boundary conditions have the form

$$A_1(x,y)u_x + A_2(x,y)u_y + A_3(x,y)u = g(x,y)$$

We use the bi-cubic Hermite functions as the basis. These are tensor products of the basis functions described in Section 2, namely

$$\phi_{ij}^{rs}(x,y) = \phi_i^r(x)\phi_j^s(y) \qquad (5\text{-}2)$$

where the functions $\phi_i^r(x)$ are defined in Section 2. The approximate solution has the representation

$$U(x,y) = \sum_{i,j=0}^{n} \sum_{r,s=1}^{2} c_{ij}^{rs}\, \phi_{ij}^{rs}(x,y) \qquad (5\text{-}3)$$

We have assumed a square N×N mesh only for simplicity of exposition. Note that there are now four basis functions associated with each mesh point. The coefficients c_{ij}^{rs} are related to the function U(x,y) by the following

$$\begin{aligned} c_{ij}^{11} &= U(x_i,y_j) & c_{ij}^{21} &= U_x(x_i,y_j) \\ c_{ij}^{12} &= U_y(x_i,y_j) & c_{ij}^{22} &= U_{xy}(x_i,y_j) \end{aligned} \tag{5-4}$$

The collocation points are the tensor products of the points used in one dimension, that is $z_{ij}^{rs} = (x_i^r, y_j^s)$. The location of these collocation points is illustrated in Figure 3. Each mesh point is associated in a natural way with its four nearest collocation points. The block Gauss-Seidel iteration then relaxes the four basis coefficients at a given mesh point simulaneously by the requirement that the equations on the four associated collocation points be satisfied. This requires the solution of a 4×4 system of equations at each mesh point. The collocation equations can again be written in the matrix form of (2-5) except that the structure of the matrix is now more complex. We again modify our notation and represent the basis coefficients c_{ij}^{rs} by U_{ij}^{rs}. An assembly process similar to that used in the finite element method can be used to construct the collocation matrix M. Given the collocation point z_{ij}^{rs} only the basis functions at the corners of the grid cell containing the collocation point contribute to the equation at that point. Therefore we need assemble only over those four corner points. The collocation equation at the point z_{ij}^{rs} can be written as follows.

$$\sum_{(k,\ell)} \sum_{p,q=1}^{2} [a_1(z_{ij}^{rs})\phi_{k\ell,xx}^{pq} + a_2(z_{ij}^{rs})\phi_{k\ell,yy}^{pq} + \ldots$$

$$a_5(z_{ij}^{rs})\phi_{k\ell}^{pq}]\, U_{k\ell}^{pq} = f(z_{ij}^{rs}) \tag{5-5}$$

The indices (k,ℓ) range over the four corner points of the cell containing z_{ij}^{rs}. In terms of the matrix M of the collocation equations this is written

$$\sum_{(k,\ell)} \sum_{p,q} m(i,j,r,s;k,\ell,p,q)\, U_{k\ell}^{pq} = f(z_{ij}^{rs}) \qquad (5\text{-}6)$$

As mentioned above, the block relaxation requires the solution of a 4×4 system of equations which involves the four collocation points associated with the grid point where the solution is updated.

The projection operator used to map the residue from the h mesh to the course 2h mesh is the tensor product of the one dimensional projection. In one dimension we denote the residue at the x_k^p collocation point by r_k^p. Note that $p\ (1 \le p \le 2)$ selects the two collocation points x_k^p associated with the $k^{\underline{th}}$ grid point. We also use subscripts c and f to denote course and fine mesh values, so that $r_{c,k}^p$ and $r_{f,k}^p$ denote residue on the course and fine mesh. Then the projection operator in one dimension can be written

$$r_{c,i}^s = \sum_{k,q} d_k^p\, r_{f,k}^p \qquad (5\text{-}7)$$

Here the indices k and q determine the four collocation points x_k^p on the fine mesh that are closest to the point x_i^s on the course mesh. The evaluation of the coefficients d_k^p is illustrated in (3-7) and (3-8). The projection mapping for the residue in two dimensions $\hat{I}_h^{2h}$ is defined by the relation

$$r_{c,ij}^{st} = \sum_{k,\ell,p,q} d_k^p\, d_\ell^q\, r_{f,k\ell}^{pq}$$

6. Results of a two dimensional computation

We applied the multigrid-collocation method to the following problem

$$a u_{xx} + a u_{yy} + a_x u_x + a_y u_y = f \qquad 0 \le x, y \le 1 \qquad (6\text{-}1)$$

The coefficient function was defined by $a(x,y) = e^{-x-y}$ and the right side $f(x,y)$ was choosen so that the solution was $u(x,y) = \sin \pi(x+y)$. Dirichlet boundary conditions were imposed on all sides of the unit square. We tested mesh resolutions of N=5, 9, and 17. The results displayed below are for N=17. A fixed mesh refinement was used with one iteration at level 4 (N=17), 2 iterations at the remaining levels 3,2, and 1. The cost for this scheme is 2.3 work units per cycle. The norm used to measure the difference between iterates involved all four values at the grid points, that is

$$\| U^{\nu+1} - U^{\nu} \|_D = [\sum_{i,j} \sum_{s,t} (U_{ij}^{st})^2/4(N+1)^2]^{1/2} \qquad (6\text{-}2)$$

The error is measured by $\|U^{\nu} - u\|_D$ and also by $\|U^{\nu} - u\|$. The former involves the derivatives, that is

$$\|U^{\nu}-u\|_D = [(\sum_{i,j} (U_{ij}^{11} - u_{ij})^2 + (U_{ij}^{21} - u_{ij,x})^2 + \ldots)/4(N+1)^2]^{1/2} \qquad (6\text{-}3)$$

The latter involves only the values of the function

$$\|U^{\nu}- u\| = [\sum_{ij} (U_{ij}^{11} - u_{ij})^2/(N+1)^2]^{1/2} \qquad (6\text{-}4)$$

Note that the error is measured relative to the solution $u(x,y)$ of the differential equation and not relative to the solution of the collocation equations. Therefore this error will not approach zero as the number of iterations grows. However, we do not seem to have reached the limiting error after 12 cycles. The results are shown in Table II below. The average convergence rate per work unit over 12 cycles for the difference between iterates is 0.65, and 0.64 for the error based on the full norm $\|U^{\nu} - u\|_D$. The convergence rate for the function values alone $\|U^{\nu} - u\|$ is 0.71. The convergence rate measured over

cycles 6 to 9 and 12 ranges from 0.68 to 0.72.

A model analysis based on Fourier series similar to the one dimensional case yields a theoretical convergence rate of 0.41. This value is based on the spectral radius of the 4×4 matrix defined by the block iteration scheme. These eigenvalues were computed numerically by an IMSL library routine. The discrepency between estimated and observed convergence rates is too large. We are currently attempting to discover an explanation. The computations described in Tables I and II were run on a DEC-10 computer with a 36 bit word length.

7. Computational efficiency

In this section we will provide an approximate operational count for the multigrid-collocation method. An estimate of storage requirements will also be given. The evaluation of the collocation matrix (5-6) requires an evaluation of the basis functions (5-2). Our code did not take advantage of the tensor product form of the basis functions, which made the computation far more costly than it should have been. The evaluation of the basis functions at each mesh point requires computation at four collocation points, four neighbors of each collocation point, five derivatives ($\phi_{xx}, \phi_{yy}, \phi_x, \phi_y, \phi$), for each of four basis functions. Each basis function requires about two floating operations and two function calls which we assume require the equivalent of eight floating operations each. Therefore the evaluation requires $4\times4\times5\times4\times18 = 5760$ operations per mesh point or $5760\ (N+1)^2$ operations for one sweep thru the mesh. If we had used the tensor product formulation this would have been reduced to $198\ (N+1) + 640\ (N+1)^2$ operations. These counts include additions as well as multiplications. To compute the matrix (5-6) given the basis functions requires five

multiplications and additions for the five derivatives, four collocation points, four neighbors, and four basis functions at each mesh point. This totals to $640(N+1)^2$ operations. Within the Gauss-Seidel iteration, the computation of the right side of the system of four equations requires a sum of products over four collocation points, three neighbors, and four basis functions for a total of $96(N+1)^2$ operations. The forward substitution requires $36(N+1)^2$ operations and the back substitution $28(N+1)^2$ operations. Note that the projection from the fine to the course mesh requires the computation of the collocation matrix (5-6) on both the fine and course mesh and the evaluation of the projection coefficients (5-7) on the course mesh. To store a variable over all the mesh refinements requires about $1.5\times4(N+1)^2$ locations, since $1 + (1/2)^2 + (1/4)^2 + \ldots = 1.5$. Therefore to store the collocation matrix (5-6) would require $1.5\times64(N+1)$ locations and to store the reduction coefficients (5-7) would require $.5\times64(N+1)$ locations. This is likely to be more storage than most applications can afford unless the collocation approximation permits a much smaller number of mesh points than a fourth order finite difference scheme would require.

To check out these operational counts we timed the code on the CDC6400 at the University of Colorado. We used the MNF compiler for the timing runs. On simple loops involving efficient subscripts for two and three dimensional arrays this compiler will usually require around 6-12 microseconds per floating point operation. This time includes the overhead for loops, subscripts, etc. However, the loops should contain no complex subscript computations or branches if this time is to be achieved. The FTN compiler will usually

run about 5 microseconds per floating point operation on such loops. The total operations required for computation and forward substitution for the matrix (5-5) is

$$(5760+640+36)\quad (N+1)^2$$

If we use 10 microseconds per floating point operation, this gives a time of 64 m.s. per mesh point. If we had coded a tensor product formulation of the basis functions this would have been 13.2 m.s. To perform the Gauss-Seidel iteration requires an additional 1.2 m.s./point. The clock times for N=16 were 57 m.s. for the matrix and 5.5 m.s. for the relaxation. The poor agreement for the relaxation portion of the code is probably due to subscript computation. The inner portion of the Gauss-Seidel loop contains two floating point operations, but 10 arithmetic operations and two array references associated with subscripts. Also a call to a linear equation solver is made at each mesh point to solve a 4×4 matrix and this must have a very high overhead. The use of simultaneous relaxation instead of Gauss-Seidel would permit the relaxation to be written in an efficient vector form free from subroutine calls and slow subscript calculation. The remainder of the code with the exception of pivoting in the linear equation solver is written in vector form and should run efficiently on the CRAY1.

In conclusion, the multigrid iteration seems to converge fairly rapidly for the collocation matrix, at least for the few cases we have run. Also, the computation is not too expensive, provided the relaxation loop is written more efficiently than we have, and provided the computation of the collocation matrix is done once and can then be amortized over many relaxation iterations. This imposes a very large storage requirement which may be too much for most problems.

Houstis, et.al [3] found the collocation method to be most efficient for elliptic problems when the matrix is solved by a direct solver. This conclusion may not hold when the matrix is solved by iteration. An iterative method may be best when the elliptic equation must be solved at each time step of a "marching problem." Such problems arise in fluid mechanics when a stream function must be computed during the solution of a time dependent marching problem. The solution of the incompressible Navier Stokes equations is such a problem. In some cases the coefficients of the elliptic equation are time dependent which causes most direct methods to become inefficient for these problems. It would also cause this iterative method to be inefficient since the matrix would have to be recomputed on each time step.

References

1. D. Boley and S. Parter, "Block relaxation techniques for finite-element elliptic equations: an example," Los Alamos Report, LA-7870, Los Alamos Scientific Laboratory (1979).

2. A. Brandt, Math. Comp., Vol. 31, No. 138, pp. 333-391.

3. E. N. Houstis, R. Lynch, J. Rice, Jour. Comp. Phys. Vol. 27, No. 3, pp. 323-350, (1978).

4. M. Schultz, "Spline Analysis," Prentice Hall, Englewood Cliffs, (1973).

5. A. Brandt, personal communication.

Cycle	Work	$\|U^{\nu+1} - U^{\nu}\|$	error e(u)	error e(u')
0	1.0	0.39	0.69	2.2
1	4.2	0.27	4.7(-2)	0.20
2	7.4	9.1(-2)	3.6(-3)	2.9(-2)
3	10.6	1.1(-2)	4.2(-4)	7.6(-3)
4	13.8	2.7(-3)	5.3(-5)	2.1(-3)
5	16.9	7.3(-4)	1.2(-5)	6.1(-4)

TABLE I.

Convergence rate for two point boundary value problem equation (4-1).

Cycle	Work	$\|U^{\nu+1} - U^{\nu}\|_D$	$\|U^{\nu} - u\|_D$	$\|U^{\nu} - u\|$
0	1.0	178.	178.	0.58
3	7.8	5.7	1.4	4.4(-2)
6	14.7	0.14	6.9(-2)	3.9(-3)
9	21.5	9.6(-3)	5.4(-3)	3.4(-4)
12	28.4	7.4(-4)	5.3(-4)	3.0(-5)

TABLE II

Convergence for the problem (6-1).

Figure 1. Projection operator I_h^{2h}

Figure 2. Projection I_h^{2h} at the boundary.

Figure 3. The collocation points on a 3×3 mesh.

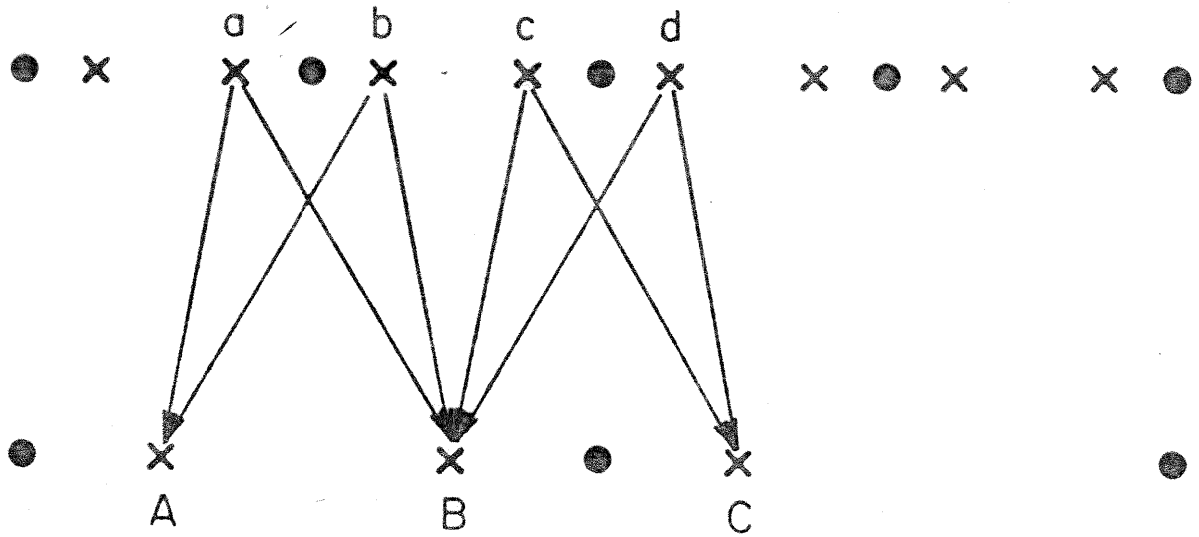

Figure 1

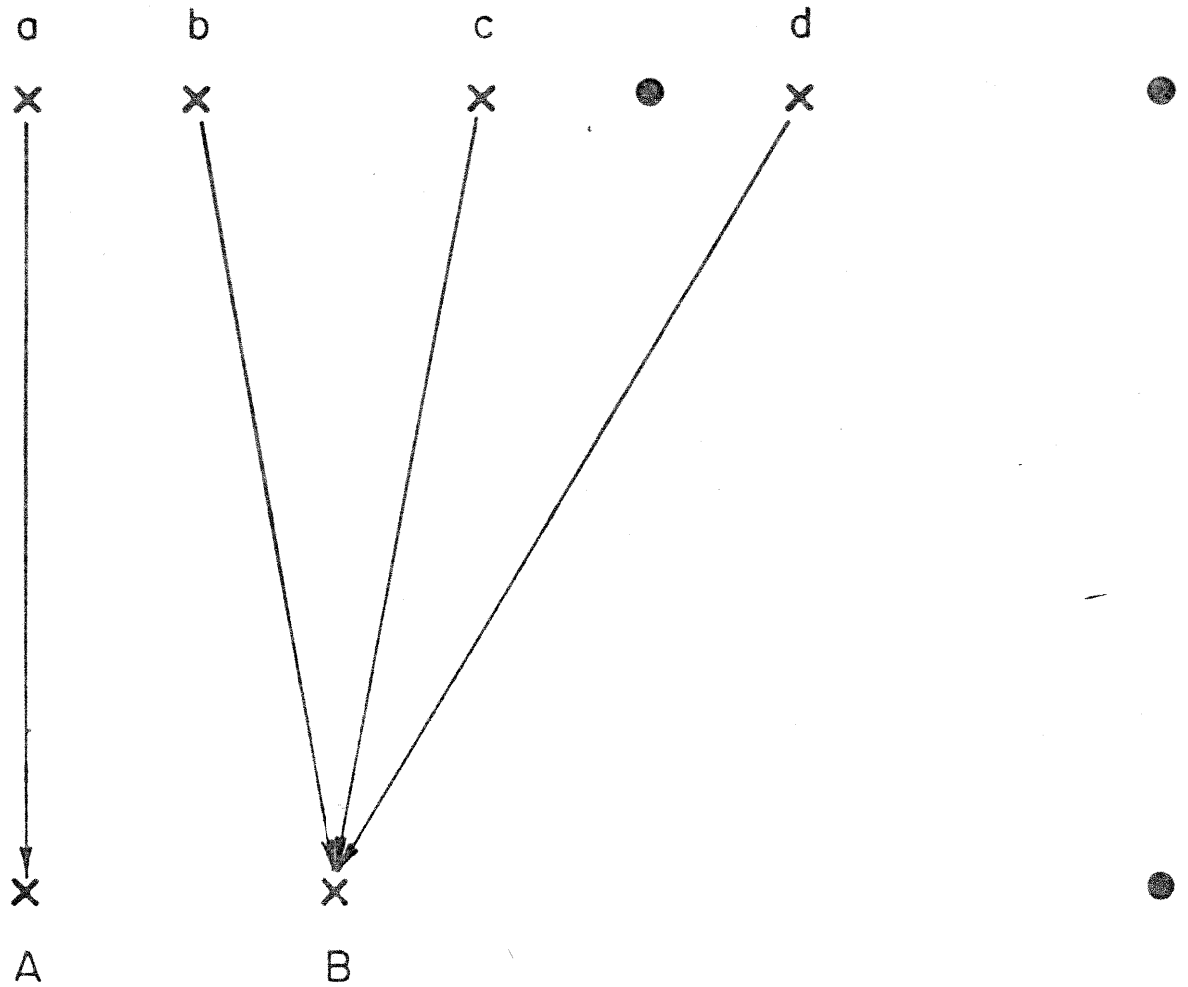

Figure 2

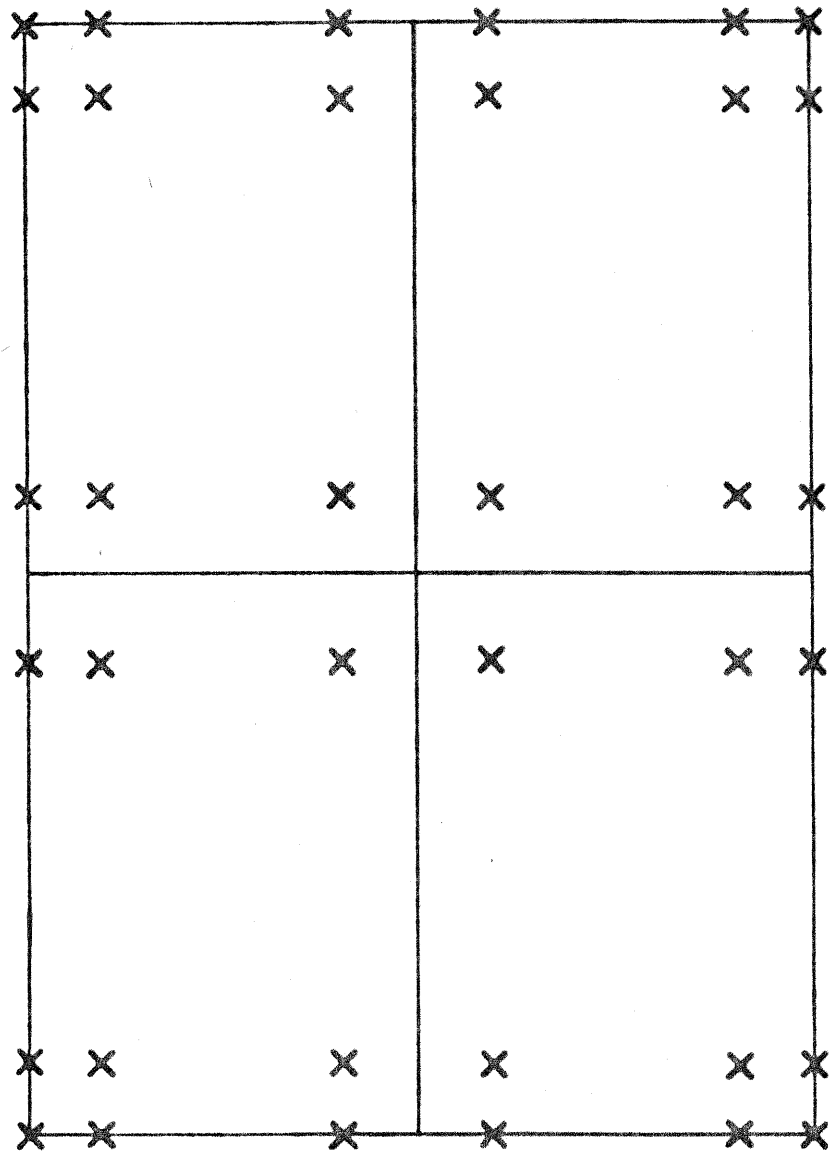

Figure 3.